Avaliação de Algoritmos para a Selecção de Vistas Materializadas em Ambientes de Data Warehousing

Filipe Martins filipe.martins@oniduo.pt
Orlando Belo obelo@di.uminho.pt
Paulo Novais pjon@di.uminho.pt

Departamento de Informática, Escola de Engenharia, Universidade do Minho
Campus de Gualtar, 4710-059 Braga
PORTUGAL

A competição no mundo empresarial obriga a uma monitorização mais apertada de todas as variáveis envolvidas nas actividades de negócio. Com o objectivo de suportar o processo de tomada de decisão em factos, e não apenas na intuição dos agentes de decisão, surgiram os sistemas de suporte à decisão. Estes sistemas são hoje uma ferramenta chave no processo de tomada de decisão, pois conciliam e integram toda a informação disponível numa única plataforma tecnológica. Assim, todas as técnicas de optimização do desempenho desses sistemas são bem-vindas. De entre as diversas técnicas disponíveis, este trabalho concentra-se na materialização de vistas como método de optimização do processamento de interrogações. A materialização de vistas consiste na antecipação do processamento e armazenamento dos tuplos resultantes do processamento da sua definição numa tabela. De facto, o tempo de reposta a uma interrogação é menor, se as operações intermédias como selecções, projecções, junções e agregações se encontrarem já armazenadas numa tabela. Desta forma, o tempo de resposta limita-se ao varrimento da vista materializada. Este artigo apresenta um estudo preliminar para o desenvolvimento de um sistema de gestão de vistas materializadas em ambientes de *data warehousing*. Neste trabalho comparam-se, basicamente, os comportamentos de dois algoritmos de selecção de vistas materializadas: o BPUS e o A*, ambos algoritmos de procura exaustiva (determinísticos).

1. Introdução

As empresas enfrentam actualmente uma crescente concorrência, potenciada pela abertura dos mercados a competidores cada vez mais ferozes. Este facto fomentou a sua competitividade levando-as a repensar e reestruturar os seus modelos de negócio. Recorrendo à reengenharia de processos, automação e informatização desenvolveram-se sistemas de informação capazes de monitorizar variáveis de produção e estimar indicadores mais precisos (em tempo real). Deste modo, colocaram-se à disposição dos agentes de decisão um conjunto de ferramentas e mecanismos de controlo, outrora impensáveis, englobados em sistemas de suporte à decisão. O aparecimento deste tipo de sistemas veio revolucionar de forma clara todo as etapas de um processo de tomada de decisão, oferecendo um acesso privilegiado à informação de maior relevo para as actividades de negócio de uma empresa.

Os sistemas de suporte à decisão têm o papel de conciliar e integrar toda a informação interna e externa à empresa numa única plataforma de dados. O objectivo primordial é disponibilizar a informação necessária aos agentes de decisão durante o processo de tomada de decisão, nas diversas vertentes do negócio. Recentemente tem-se assistido a um aumento do investimento em investigação e desenvolvimento de técnicas de optimização. Estas técnicas, como as de *data warehousing*, são utilizadas no melhoramento do desempenho de sistemas de informação, com particular incidência no aumento do desempenho dos sistemas de suporte à decisão. A materialização de vistas é uma das técnicas mais utilizadas na optimização de bases de dados e de *data warehouses*, mais concretamente na optimização do processamento de interrogações. Esta técnica assenta no conceito da antecipação do cálculo (total ou parcial) das interrogações, de forma a minimizar o impacto que os seus tempos de resposta teriam no desempenho global do sistema. A optimização do processamento de interrogações utilizando a técnica de materialização de vistas divide-se essencialmente em três áreas de estudo [1, 2]: a selecção de um conjunto de vistas a materializar; a utilização de um conjunto de vistas materializadas; a manutenção de um conjunto de vistas materializadas.

O presente estudo, é um trabalho preliminar para o desenvolvimento e construção de um sistema de gestão de vistas materializadas, em ambiente de *data warehousing*. Neste contexto interessa investigar e perceber a utilidade (vantagens e desvantagens) das vistas materializadas. Como primeiro passo, optou-se por estudar o problema da selecção de vistas materializadas em ambientes de *data warehousing* e em particular, o estudo e compreensão do comportamento de alguns algoritmos para a sua selecção. Na secção seguinte aborda-se a temática da materialização de vistas no contexto da optimização do processamento de interrogações em ambiente de base de dados multidimensionais. Após isto, dedicamos uma secção sobre algoritmos de selecção, na qual se apresenta uma classificação acerca desses algoritmos, uma definição formal do problema da selecção de um conjunto de vistas a materializar, uma simplificação ao modelo de custo, os critérios de escolha dos algoritmos testados e uma breve descrição das propriedades dos mesmos algoritmos. Mais adiante testam-se e fazem-se as análises críticas dos resultados obtidos com os dois algoritmos de procura exaustiva escolhidos, o BPUS e o A*. Por fim, termina-se com uma secção de conclusões críticas, na qual se deixam algumas sugestões e apontam novos rumos a serem, eventualmente, seguidos em trabalhos futuros.

2. Materialização de Vistas

A materialização de um conjunto de vistas é uma técnica sobejamente utilizada em base de dados, principalmente em *data warehousing*. Esta última é uma consequência natural do refinamento da ideia de se materializarem vistas, particularmente, o modelo multidimensional é em última análise uma implementação de vistas (após alguma normalização de dados). Nesta secção apresentam-se alguns conceitos sobre vistas materializadas, no âmbito da optimização em ambiente de *data* warehousing, ou seja na optimização do processamento de interrogações sobre base de dados multidimensionais.

Materializar uma vista numa base de dados consiste em armazenar os tuplos resultantes do processamento da sua definição numa tabela [2]. Essa tabela é denominada por vista materializada. Alguns autores [3] consideram que um *data warehouse* não é mais que um conjunto de vistas materializadas, onde se armazenam os dados provindos das fontes de informação (ex. sistemas operacionais). Teoricamente se a resposta a uma interrogação for antecipadamente processada e armazenada numa tabela, o tempo de resposta restringir-se-ia ao tempo de varrimento dos tuplos dessa tabela. Este tempo de varrimento pode ainda ser encurtado utilizando estruturas auxiliares de indexação [4-6]. Portanto, qualquer interrogação cujo plano de execução possa ser reescrito de forma a utilizar o conjunto vistas materializadas (disponível na base de dados) está sujeito a um ganho no desempenho. Para interrogações complexas envolvendo grandes volumes de dados os ganhos possíveis utilizando vistas materializadas são dramáticos, de horas ou dias para segundos ou minutos. De facto, as vistas materializadas são tidas como sendo uma das técnicas essenciais para aumentar o desempenho de um *data warehouse* [3-5, 7-15]. Assim, e com vista à optimização do sistema, o ideal seria armazenar as respostas para todas as interrogações possíveis. Contudo, armazenar todas as vistas e agregações possíveis para um dado esquema requer espaço em disco e tempo de processamento.

Neste contexto surge um dos desafios que se coloca à utilização de vistas materializadas. Trata-se da selecção de um conjunto de vistas que minimize o tempo de processamento de interrogações dadas as restrições de espaço e de tempo de processamento das vistas. Esta questão é conhecida na literatura pelo problema da selecção de vistas a materializar.

As tabelas base, assim denominadas as relações do esquema da base de dados sobre as quais se definem as vistas, sofrem várias actualizações, inserções e remoções com o decorrer do tempo. Estas alterações devem reflectir-se nas vistas materializadas para manter a sua consistência com as tabelas base. A inconsistência dos dados entre as vistas materializadas e as tabelas base tem um custo elevado no processamento de interrogações. Isto porque a resposta a uma interrogação tem de ser a mesma, quer esta seja lançada sofre o esquema da base de dados, quer sobre o conjunto de vistas materializadas (isto se a interrogação puder ser reescrita em termos das vistas materializadas disponíveis). Este é conhecido na literatura pelo problema da manutenção de vistas materializadas.

A utilização das vistas materializadas é também uma das questões de relevante importância sob investigação. Os utilizadores lançam interrogações escritas em termos do esquema da base de dados. Isto porque desconhecem a existência das vistas materializadas ou porque o dinamismo do sistema obrigaria a uma constante actualização do conhecimento relativamente ao conjunto de vistas. De notar que os utilizadores da base de dados podem também ser aplicações, que possuem um conjunto de interrogações predefinidas (por exemplo, para elaboração de relatórios). Estas aplicações podem ser incapazes de se adaptar a alterações ocorridas no conjunto de vistas materializadas. A ideia é tornar a utilização de vistas materializadas num mecanismo invisível aos utilizadores. Ou seja, as interrogações lançadas em termos do esquema da base de dados têm de ser reescritas em termos das vistas disponíveis sempre que tal seja possível e que beneficie o desempenho do sistema. Este é o problema da reescrita de interrogações atendendo a um conjunto de vistas materializadas disponível.

Na figura1 ilustra-se um esquema representativo da trilogia de questões que se levantam aquando o uso da técnica de materialização de vistas para optimizar o desempenho do processamento de interrogações.

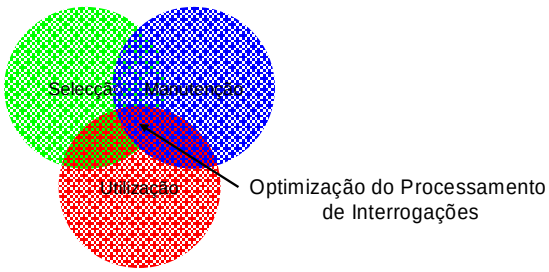

Figura 1 – Vistas materializadas e a optimização do processamento de interrogações

A materialização de vistas é uma técnica valiosa na optimização do processamento de interrogações. Pode originar ganhos elevados no tempo de resposta a interrogações e, com isto, aumentar claramente o desempenho do sistema. Contudo, é uma técnica que consome "alguns" recursos em termos de armazenamento e tempo de processamento. É preciso encontrar-se um meio-termo entre os ganhos de desempenho e os recursos disponíveis. Este balanceamento é o alvo do estudo do problema da selecção de vistas. A materialização de vistas obriga ainda a acautelar problemas de inconsistência de dados, abordados na investigação do problema da manutenção de vistas. Outro ponto crucial está em encontrar-se meios de tornar esta técnica num mecanismo invisível aos utilizadores e aplicações informáticas (problema da utilização de vistas). O problema da selecção de vistas não é novo [16] e tem sido sobejamente abordado na literatura. Contudo, as investigações e estudos do problema pecam por serem demasiadamente teóricos. Neste trabalho pretende-se estudar o problema da selecção de vistas materializadas em ambientes de *data warehousing* de uma forma prática. Pretende-se também implementar e avaliar algoritmos de selecção. Este é o primeiro passo com vista à concepção de um protótipo para a materialização de vistas em sistemas de *data warehousing*. A maioria das abordagens ao problema da selecção de vistas a materializar apresentadas na literatura incidem em algoritmos de pesquisa sobre um espaço de soluções possíveis. Estes algoritmos têm como finalidade devolver o melhor conjunto de vistas a materializar, dadas um conjunto de interrogações frequentes e alguma restrição (espaço ou tempo).

3. Algoritmos para a Selecção de Vistas

Tipologia de algoritmos

Segundo *Zhang Chuan* [13] os algoritmos de selecção caracterizam-se em quatro tipos: determinísticos, aleatórios, genéticos e híbridos. Na primeira categoria encon-

tram-se os algoritmos que devolvem uma solução de forma determinística pela aplicação de heurísticas ou por procura exaustiva. Os algoritmos aleatórios regem-se por uma abordagem muito diferente. No início são definidos movimentos entre as diversas soluções pertencentes ao espaço de soluções do problema. Estes algoritmos percorrem aleatoriamente as soluções de acordo com regras predefinidas à partida, terminando assim que não forem possíveis mais movimentos ou se esgotar um tempo limite. A melhor solução encontrada é então devolvida. A escolha das regras que determinam os movimentos tem um impacto significativo no desempenho destes algoritmos [17]. Na terceira categoria agrupam-se os algoritmos que utilizam uma estratégia aleatória muito semelhante à evolução biológica na procura da melhor solução, daí o nome de algoritmos genéticos. Neste tipo de algoritmos parte-se de uma população inicial aleatoriamente gerada. Em cada geração são efectuados cruzamentos e mutações. Somente os membros mais adaptados da população passam para próxima geração de acordo com uma função de custo. Estes algoritmos terminam quando deixam de haver melhorias significativas nas novas gerações, ou após um certo número de gerações. O melhor cromossoma (ou indivíduo) encontrado na população final é a solução. Finalmente, na categoria de algoritmos híbridos encontram-se aqueles que utilizam uma mistura de algoritmos puramente determinísticos com algoritmos aleatórios. Usualmente as soluções obtidas através de algoritmos determinísticos são utilizadas como pontos de partida (sementes) para algoritmos aleatórios, ou povoações iniciais para algoritmos genéticos.

Definição do problema

Seja L (um grafo acíclico por exemplo) o espaço de soluções e M ($M \subseteq L$) um conjunto de vistas a materializar. Cada vista $v \in L$ tem três valores associados: a frequência de interrogação f_v, a frequência de actualização g_v e o custo de leitura r_v. Segundo [17] o custo de responder a uma interrogação q (que corresponde a uma vista v) corresponde ao tamanho de v, ou seja ao número de tuplos (r_v). O problema da determinação de um espaço L de soluções é abordado na literatura em [18]. Nesta fase apenas se atende ao problema da selecção do conjunto de vistas M a materializar pressupondo a existência de L. O custo de armazenamento de M é dado por:

$$S(M) = \sum_{v \in M} r_v \ . \qquad \textbf{(1)}$$

Seja $u(v, M)$ o custo de actualização da vista v quando M é o conjunto de vistas materializadas. Então o custo de actualização de M é dado por:

$$U(M) = \sum_{v \in M} g_v \times u(v, M) \ . \qquad \textbf{(2)}$$

Seja $q(v, M)$ o custo de se responder a uma interrogação v quando M é o conjunto de vistas materializadas. O custo total de responder a interrogações é dado por:

$$Q(M) = \sum_{v \in L} f_v \times q(v, M) \ . \qquad \textbf{(3)}$$

O problema da selecção de vistas a materializar consiste em escolher um conjunto M tal que $Q(M)$ é mínimo e que se respeitem as restrições $S(M) \leq S_{\max}$ e $U(M) \leq U_{\max}$. Sendo $S_{\max}$ o espaço disponível para materialização e $U_{\max}$ a janela temporal disponível para a manutenção do conjunto de vistas materializadas.

O modelo de custo utilizado

O modelo de custo utilizado neste trabalho sofreu uma pequena simplificação relativamente ao apresentado anteriormente na definição do problema. O modelo de custo tem apenas em consideração o custo de processamento de interrogações sobre o conjunto de vistas materializadas, desprezando os custos de manutenção. Isto deve-se ao facto de um *data warehouse* não sofrer actualizações [19]. Sendo assim, o problema limita-se a ter apenas uma restrição, a restrição de espaço, $S(M) \leq S_{\max}$. E o custo de se responder a uma interrogação utilizando um determinado conjunto de vistas materializadas, $q(v, M)$, consiste no número de tuplos das vistas ou tabelas envolvidas no processamento da resposta [2, 20-22].

Algoritmos seleccionados

A selecção dos algoritmos implementados (e testados) regeu-se fundamentalmente por três critérios. Escolheram-se algoritmos do mesmo tipo para posterior análise e comparação de resultados. Como segundo critério, escolheram-se algoritmos do tipo determinísticos porque, e segundo a literatura[1-5, 20-22], estes garantem soluções próximas da óptima. O último assentou na credibilidade demonstrada através do número de referências na literatura. Assim, com base nos critérios enunciados, seleccionaram-se os seguintes algoritmos: o BPUS e o A*.

O algoritmo BPUS[1] apresentado por *Himanshu Gupta* em 1997 [3, 4, 21, 22] é dos mais discutidos e estudados. Utiliza grafos acíclicos orientados para representar as vistas e interrogações, sendo que cada nó de um grafo G representa uma operação de agregação, selecção, projecção ou junção. A cada nó está associado um tamanho (número de tuplos) e uma frequência de interrogação (número de vezes que o nó foi acedido). O algoritmo, em cada iteração, selecciona a vista com o máximo benefício por unidade de espaço. A noção de benefício de uma vista v em relação a conjunto de vistas materializadas M, foi uma das novidades apresentadas pelo autor, e é calculado através da equação 5.

$$t(G, M) = \sum_{i=1 \wedge vi \in G}^{k} f_{vi} q(vi, M) \qquad (4)$$

O cálculo do benefício de uma vista consiste na diferença dos custos de processamento do conjunto de vistas materializadas com e sem a vista. O custo total de um conjunto de vistas materializadas é dado pela equação 4.

[1] *Benefit per unit of space*

$$B(v, M) = t(G, M) - t(G, M \cup \{v\}). \quad (5)$$

O benefício por unidade de espaço (equação 6) é calculado através do quociente entre o benefício apresentado pela vista *v* e o seu espaço, $S(v)$.

$$BPUS(v, M) = \frac{B(v, M)}{S(v)} \quad (6)$$

Segundo o autor o tempo de processamento do BPUS é $O(kn^2)$, onde *n* é o número de nós no grafo *G* e *k* é o número de iterações utilizado pelo algoritmo 1.

```
Dados: G é um grafo de vista ANDOR, S uma restrição de espaço

INICIO
   M = {};
   Enquanto(S(M)<S)
      Seja V a vista que tem um máximo benefício
          por unidade de espaço em relação a M.
      M = M U {V};
   Fim de Enquanto
   Devolve M
FIM
```

Algoritmo 1 – O algoritmo BPUS

```
Dados: Um cubóide G e uma restrição de espaço S.
INICIO
   Criar uma árvore de pesquisa A* inicial TG,
      tendo como nó de raíz ({}, {});
   Determinar uma ordem de inserção de vistas:
      (v1, v2, ..., vn);
   Criar uma lista de prioridade L = {raíz};
   Repete
      Retirar um nó x = (Nx, Mx) de L, onde x
         tem o menor g(x) + h(x) valor em L;
      i = |Nx|;
      Se (i = n) Então
         Devolve Mx;
      Fim de Se
      Inserir l(x) = (Nx U {vi+1}, Mx) em L;
      Se (U(Mx U {vi+1}) <= S) Então
         Inserir r(x) = (Nx U {vi+1}, Mx U {vi+1})
            em L;
      Fim de Se
   Até(L estar vazia);
   Devolve {};
FIM
```

Algoritmo 2 – O algoritmo A*

O segundo algoritmo, A* (algoritmo 2), apresentado em [23] para a selecção sob uma restrição de tempo de manutenção, e estendido por [20] para a selecção sob uma

restrição de espaço, utiliza igualmente os grafos acíclicos orientados como forma de representação do problema. Contudo em [20] especializa-se a representação para um cubóide, que não é mais que um grafo acíclico orientado adaptado ao modelo multidimensional, com um nó como raiz e outro como folha. O nó de raiz é a tabela de factos e a folha consiste na agregação máxima (ou universal) dos dados da tabela de factos, ou seja o total sobre todas as dimensões. Mais adiante apresenta-se um exemplo de um cubóide.

Para um dado cubóide *G*, o algoritmo A* expande uma árvore binária de procura *TG*. Cada nó em *TG* é do tipo (Nx, Mx), denotado por $x = (Nx, Mx)$, onde Nx é o conjunto de vistas visitadas e Mx é o conjunto de vistas seleccionadas para materialização $(Mx \subseteq Nx)$. Seguindo a ordem de inserção predeterminada $(v_1, v_2, ..., v_n)$, cada vista v_i é considerada para materialização no *i*-gésimo nível de *TG*. O descendente esquerdo de x é definido por $l(x) = (N_x \cup \{v_{i+1}\}, M_x)$, que significa que a recém inserida vista v_{i+1} não será materializada. O descendente direito *x* definido por $r(x) = (N_x \cup \{v_{i+1}\}, M_x \cup \{v_{i+1}\})$ significa que a recém inserida vista v_{i+1} será materializada. Finalmente, no nível mais baixo da árvore $N_x = V$, o que significa que todas as vistas do cubóide *G* foram consideradas ou não para materialização (*V* denota o conjunto de vistas ou nós em *G*). O benefício de expandir o nó *x* é a soma de duas funções: *g(x)* e *h(x)*. A primeira calcula o benefício adquirido e a segunda o benefício estimado ao materializar *x* (ver [20]). Segundo *Gang Gou et al.* [20] o processo de construção da árvore binária de procura A* cobre todo o espaço de 2^n soluções possíveis. Isto assegura que o algoritmo contempla a solução óptima. A complexidade do algoritmo A* situa-se entre *O(n)* e $O(2^n)$ dependendo da qualidade da função estimadora *h(x)*. Ou seja, no pior dos casos tem um comportamento exponencial, muito pior que o BPUS.

4 Análise e comparação dos algoritmos BPUS e A*

Num *data warehouse* os dados são organizados segundo um modelo multidimensional, de forma a melhorarem o desempenho do processamento de dados e a sua visualização. Utiliza-se aqui o mesmo exemplo apresentado em [20]. Neste exemplo, tem-se um esquema com três dimensões e uma tabela de factos com quatro atributos: `VENDAS(Tempo, Produto, Loja, Preço)`. Os atributos `Tempo (T)`, `Produto (P)` e `Loja (L)` formam o espaço dimensional. O último atributo, `Preço`, é a medida sobre a qual se pretendem analisar os dados. As análises feitas em OLAP consistem em agregações das medidas por algumas dimensões. As funções[2] de agregação usualmente disponíveis em base de dados, juntamente com o operador `GROUP-BY`, são: a soma; contagem; média; máximo: mínimo. As vendas de produto por loja traduzem-se na seguinte interrogação SQL: `SELECT Produto, Loja, SUM(Preço) FROM VENDAS GROUP BY Produto, Loja`. Nesta interrogação recorre-se ao tipo de agregação produto por loja (`PL`).

[2] Conjunto de funções de agregação = {SUM, COUNT, AVG, MAX, MIN}

No esquema multidimensional apresentado podem efectuar-se 2^3 tipos de agregações diferentes (PLT, PL, PT, LT, P, L, T). E considerando também a agregação das medidas num único total, que neste trabalho se designará por agregação universal (ALL). Geralmente recorre-se a um cubóide para representar os relacionamentos entre as agregações possíveis de se efectuarem sobre um dado esquema multidimensional. Por exemplo, a partir da agregação produto por loja (PL) é possível ainda efectuarem-se mais duas agregações (por produto, P, e por loja, L). Destas duas pode-se ainda chegar à agregação universal (ALL). Em [20] encontra-se uma definição detalhada de cubóide. Na figura 2 ilustra-se o cubóide do exemplo utilizado, onde TF simboliza a tabela de factos. Para cada nó do cubóide existem dois parâmetros, T e f. O primeiro indica o número de tuplos que resultam da agregação especificada. O segundo indica a frequência de interrogações que utilizam a agregação em causa. Note-se que qualquer interrogação pode ser respondida a partir da tabela de facto, motivo pela qual esta não tem uma frequência de interrogação associada.

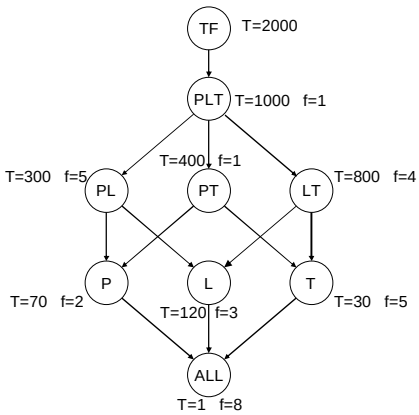

Figura 2. Cubóide

Os algoritmos foram implementados em JAVA[3]devido à portabilidade e modularidade que a linguagem confere. Ambos os algoritmos foram executados num ciclo onde se fez variar a restrição de espaço. O espaço necessário para materializar todas as vistas do cubóide é de 2721 (somatório do número de tuplos de todas as agregações ou vistas). De seguida, apresentam-se alguns gráficos resultantes de um primeiro teste. O primeiro gráfico relaciona o custo total de processamento das soluções obtidas por ambos os algoritmos, com a evolução da restrição de espaço (fig.3). Note-se que quanto o espaço de restrição é nulo (ou seja, o conjunto de vistas a materializar é vazio) os custos de processamento são de 58000. Já quando se atinge uma restrição de espaço de 2721, o custo de processamento é mínimo (6758). Note-se uma anomalia no algoritmo A* que verifica uma degradação no custo de processamento quando a restrição de espaço atinge o valor 301. A solução anterior materializa o conjunto de

[3] JAVA™ 5.0 (“Tiger”) da Sun Microsystems™.

vistas *M={ALL, T, L, P}* e tem um custo de 22658, quando a restrição passa de 300 para 301, o A* selecciona o conjunto *M={ALL, PL}* com um custo de processamento de 25008. Tal sucede porque o algoritmo dá preferência às vistas com maior frequência de interrogação e tamanho. É então necessário implementar uma outra função estimadora *h(x)*, que não está nos objectivos deste estudo, e por isso fica como proposta para trabalhos futuros. Observando a figura 3 poder-se-ia afirmar que o BPUS tem um melhor comportamento que o A*, para o exemplo em causa. Contudo, o gráfico da fig. 4 evidencia que o BPUS não respeita a restrição de espaço imposta. Na verdade, o algoritmo implementado materializa sempre uma vista extra, e como tal ultrapassa o limite de espaço imposto pela restrição. Isto reflecte-se no comportamento do algoritmo, que acaba por apresentar melhor desempenho que o concorrente, uma vez que tira partido de espaço que na realidade não possui.

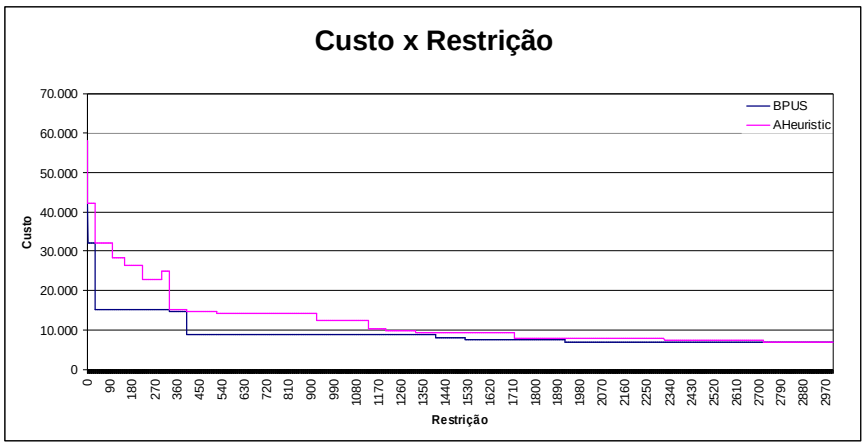

Figura 3. Custo *versus* Restrição (primeiro teste)

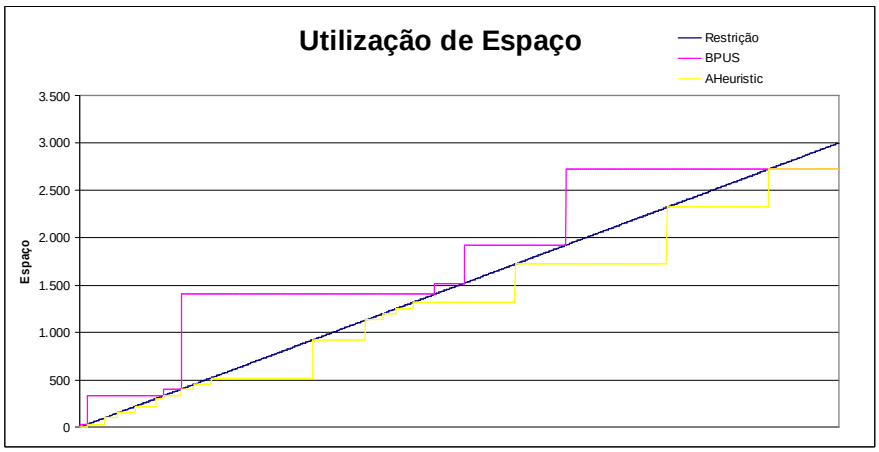

Figura 4. Utilização de espaço (primeiro teste)

Procederam-se a alguns ajustes no funcionamento do BPUS, de forma a contemplar na íntegra a restrição de espaço, o que o torna mais realista. Desta feita, e após novos testes verifica-se que o BPUS alterado (figura 6) respeita a restrição de espaço.

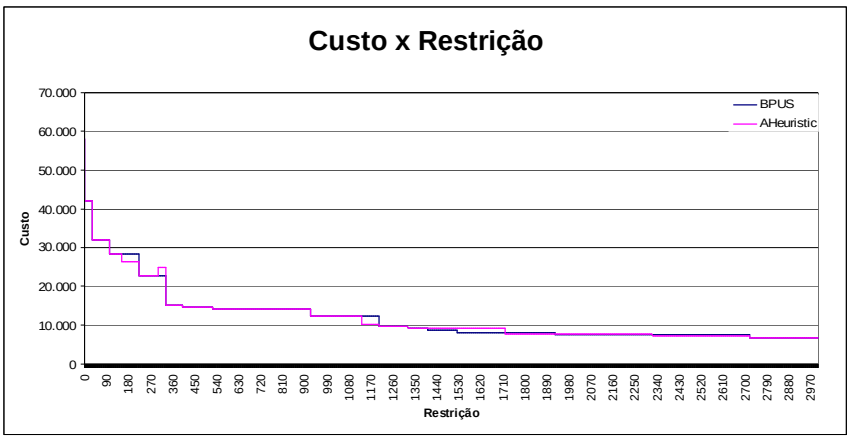

Figura 5. Custo *versus* Restrição (segundo teste)

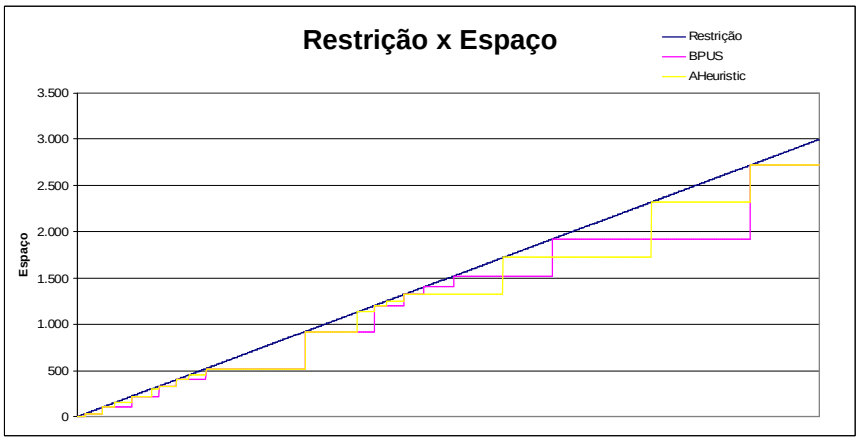

Figura 6. Utilização de espaço (segundo teste)

Os resultados do segundo teste evidenciam um comportamento semelhante dos dois algoritmos para o exemplo utilizado (figura 5). Isto mostra que no primeiro teste o BPUS tirou claramente partido da vista extra que podia materializar, para obter ganhos em termos de custo de processamento. Efectuou-se ainda uma última análise em termos do tempo dispendido por cada algoritmo na procura da solução, fazendo igualmente variar a restrição de espaço. A figura 7 mostra que o A* tem melhor comportamento, quando comparado com o BPUS.

Destas análises podem concluir-se duas coisas. O BPUS, depois de alterado, fornece soluções mais consistentes que o A*, que por sua vez mostra ser mais rápido a

resolver os mesmos problemas. A segunda ideia é que em última análise os algoritmos têm um comportamento parecido, o que provavelmente terá a ver com o seu tipo (ambos determinísticos). Para confirmar esta tese seria necessário verificar com a implementação de outros algoritmos do mesmo tipo [24], se haveria alterações significativas nos seus comportamentos.

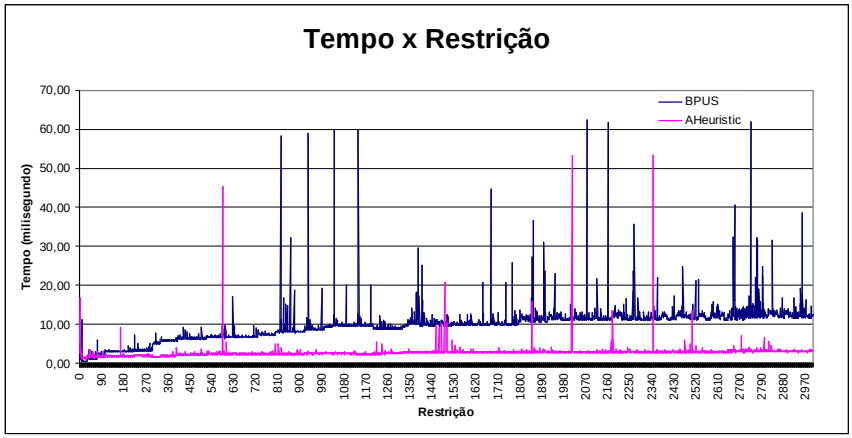

Figura 7. Tempo *versus* Restrição

4. Conclusões e Trabalho Futuro

Nos dias que correm, a premência em informação útil e disponibilizada atempadamente motivou o aparecimento dos sistemas de suporte à decisão. Bem como o aparecimento de técnicas de optimização para aumentar o desempenho desses sistemas. A materialização de vistas como técnica de optimização do processamento de interrogações, em ambiente de *data warehousing*, é teoricamente um bom caminho. Com este trabalho pretendeu-se dar o primeiro passo numa série de etapas para a construção de um sistema de gestão de vistas materializadas. Os algoritmos de selecção BPUS e A* são os mais invocados na literatura, e como tal foram escolhidos para análise. Os resultados obtidos mostram que o BPUS tem um melhor comportamento que o A* devido à forma como foi concebido. A sua concepção lógica permite-lhe implementar uma vista extra, desrespeitando a restrição de espaço. Contudo, depois de modificado de forma a restringir-se ao parâmetro de espaço disponível, verifica-se que o seu comportamento, em termos de custo de processamento, é semelhante ao do seu concorrente. Por sua vez o A* apresenta-se mais rápido para encontrar uma solução. Contudo tem uma discrepância na evolução do custo de processamento com a restrição de espaço. Tal deve-se ao facto de não se ter encontrado uma função estimadora *h(x)* apropriada. Fica no ar a ideia de testar-se em futuros trabalhos uma simbiose entre estes dois algoritmos, aproveitando as vantagens de cada um. Ficou demonstrado com este trabalho que ambos os algoritmos apresentam soluções idênticas, faltando diversificar os testes em termos de dimensão do problema (por exemplo, aumentar o número de dimensões da tabela de factos) com o intuito de concluir algo quanto à escalabi-

lidade dos algoritmos. Ainda no campo do que ficou por fazer, e para além dos testes de escalabilidade, faltou ainda confirmar com a implementação de outros algoritmos, se o comportamento é idêntico para todos os algoritmos determinísticos. Este teste poderia então ser estendido aos restantes tipos de algoritmos: aleatórios; genéticos; híbridos.

Referências

1. Rada Chirkova, A.Y.H., Dan Suciu, A Formal Perspective on the View Selection Problem, in VLDB. 2001. p. 59-68.
2. Ashish Gupta, I.S.M., Maintenance of Materialized Views: Problems, Techniques, and Applications, in IEEE Data Eng. Bull. 1995. p. 3-18.
3. Himanshu Gupta, I.S.M., Selection of Views to Materialize in a Data Warehouse, in IEEE Transcations on Knowledge and Data Engineering. 2005. p. 24-43.
4. Himanshu Gupta, V.H., Anand Rajaraman, Jeffrey D. Ullman, Index Selection for OLAP, in ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering. 1997: Washington, DC, USA. p. 208--219.
5. Venky Harinarayan, A.R., Jeffrey D. Ullman, Implementing data cubes efficiently, in SIGMOD Rec. 1996: New York, NY, USA. p. 205--216.
6. Roussopoulos, N., View indexing in relational databases, in ACM Trans. Database Syst. 1982: New York, NY, USA. p. 258--290.
7. Elena Baralis, S.P., Ernest Teniente, Materialized Views Selection in a Multidimensional Database, in The {VLDB} Journal. 1997. p. 156-165.
8. Foto Afrati, R.C., Shalu Gupta, Charles Loftis, Designing and Using Views to Improve Performance of Aggregate Queries, in Tenth International Conference on Database Systems for Advanced Applications. 2005: Beijing, China.
9. Goretti K. Y. Chan, Q.L., Ling Feng, Design and selection of materialized views in a data warehousing environment: a case study, in DOLAP '99: Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP. 1999: New York, NY, USA. p. 42--47.
10. Jian Yang, K.K., Qing Li, A Framework for Designing Materialized Views in Data Warehousing Environment, in International Conference on Distributed Computing Systems. 1997. p. 0-.
11. Jian Yang, K.K., Qing Li, Algorithms for Materialized View Design in Data Warehousing Environment, in The {VLDB} Journal. 1997. p. 136-145.
12. C. Zhang, X.Y., J. Yang, An evolutionary approach to materialized view selection in a data warehouse environment, in IEEE Transactions on Systems, Man and Cybernetics, Part C. 2001. p. 282--294.
13. Chuan, Z., Y. Xin, and Y. Jian, Evolving Materialized Views in Data Warehouse, in Proceedings of the Congress on Evolutionary Computation, J.A.a.Z.M.a.M.S.a.X.Y.a.A.Z.L.z.e. Peter, Editor. 1999: Mayflower Hotel, Washington D.C., USA. p. 823--829.
14. Dimitri Theodoratos, M.B., A general framework for the view selection problem for data warehouse design and evolution, in DOLAP '00: Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP. 2000: New York, NY, USA. p. 1--8.
15. Minsoo Lee, J.H., Speeding up Warehouse Physical Design Using a Randomized Algorithm, in Design and Management of Data Warehouses. 1999. p. 3.
16. Jose A. Blakeley, P.-A., Larson, Frank Wm. Tompa, Efficiently Updating Materialized Views, in {SIGMOD} Conference. 1986. p. 61-71.
17. Panos Kalnis, N.M., Dimitris Papadias, View selection using randomized search, in Data Knowl. Eng. 2002: Amsterdam, The Netherlands, The Netherlands. p. 89--111.

18.Dimitri Theodoratos, W.X., Constructing search spaces for materialized view selection, in DOLAP '04: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP. 2004: New York, NY, USA. p. 112--121.
19.Ralph, K., et al., The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses with CD Rom. 1998: John Wiley \& Sons, Inc. 800.
20.Gang Gou, J.X.Y., Hongjun Lu, A* Search: An Efficient and Flexible Approach to Materialized View Selection, in IEEE Transactions on Systems, Man, and Cybernetics (Part C). 2004.
21.Gupta, H., Selection of Views to Materialize in a Data Warehouse, in ICDT '97: Proceedings of the 6th International Conference on Database Theory. 1997: London, UK. p. 98--112.
22.Gupta, H., Selection and Maintenance of Views in a Data Warehouse. 1999.
23.Himanshu, G. and M. Inderpal Singh, Selection of Views to Materialize Under a Maintenance Cost Constraint. Proceeding of the 7th International Conference on Database Theory. 1999: Springer-Verlag. 453-470.
24.Satyanarayana R. Valluri, S.V., Kamalakar Karlapalem, View relevance driven materialized view selection in data warehousing environment, in CRPITS '02: Proceedings of the thirteenth Australasian conference on Database technologies. 2002: Darlinghurst, Australia, Australia. p. 187--196.